WHAT IS CLAIMED IS:

1. A method of managing messages, comprising:

storing messages in a plurality of queues;

providing a macro queue associated with the plurality of queues;

calling an application programming interface to initiate a request to the macro queue to obtain a message stored in one of the plurality of queues without identifying a particular queue; and

selecting a queue from among the plurality of queues and selecting a message from the selected queue.

2. The method of claim 1, further comprising assigning a priority value to each of the plurality of queues.

3. The method of claim 2 wherein the macro queue selects a message from a queue having the highest priority value.

4. The method of claim 1 wherein the macro queue selects a message that has been stored in the plurality of queues for the longest time.

5. The method of claim 1, further comprising providing a remote queue proxy for establishing a communication link between a remote application programming interface and the macro queue.

6. The method of claim 1 wherein the plurality of queues and the macro queue are software objects that are implemented using object oriented programming principles.

7. The method of claim 6, further comprising calling a software function of the macro queue object to associate a queue object with the macro queue object, wherein the software function returns a queue instance pointer pointing to the location of the queue object and a priority value representing the priority of the queue.

8. The method of claim 6, further comprising calling a software function of the macro queue object to remove the association between the macro queue and a queue.

9. A method of managing messages, comprising:

providing an application programming interface (API) to allow a producer module to send a message to a macro queue that manages a plurality of queues, the API sending the message to the macro queue without identifying one of the plurality of queues.

10. The method of claim 9 wherein the macro queue selects the first queue that is available among the plurality of queues and sends the message to the selected queue.

11. The method of claim 9 wherein the macro queue duplicates the message and sends the message to all of the plurality of queues.

12. The method of claim 9 wherein the macro queue selects a queue from among the plurality of queues that has the fastest response time based on previous response time records and sends the message to the selected queue.

13. The method of claim 9 wherein the macro queue selects a queue by cycling through each of the plurality of queues in a round robin fashion, and sends the message to the selected queue.

14. The method of claim 9 wherein the macro queue and the plurality of queues are implemented as software objects according to objected oriented programming principles.

15. A method comprising:

keeping a list of queue pointers, each pointer pointing to one of a plurality of queues;

receiving a request for adding a queue element; and

servicing the request by selecting one or more queue pointers from the list based on a predetermined criterion and adding the queue element to the one or more queues that the selected one or more queue pointers are pointing to.

16. The method of claim 15 wherein the predetermined criterion is to select a queue pointer pointing to a queue that has the shortest response time.

17. The method of claim 15 wherein the predetermined criterion is to select all of the queue pointers.

18. The method of claim 15 wherein the predetermined criterion is to select a queue pointer from the list in a round robin fashion by cycling through each of the queue pointers in the list.

19. A method comprising:

keeping a list of queue pointers, each pointer pointing to one of a plurality of queues;

receiving a request for retrieving a queue element; and

servicing the request by selecting one or more queue pointers from the list based on a predetermined criterion and retrieving a queue element from the one or more queues that the selected one or more queue pointers are pointing to.

20. The method of claim 19 wherein the predetermined criterion is to select a queue pointer pointing to a queue that is the first one to be available.

21. The method of claim 19 wherein each of the queues has a priority value, and the predetermined criterion is to select a queue pointers pointing to a queue having the highest priority value.

22. A method for messages communication in a distributed system, comprising:

providing an application programming interface on each computer of a group of computers in the distributed system;

providing a remote queue proxy on each of the computers of the group;

initiating a request through an application programming interface on a first computer of the group; and

passing said request to a second of the computers of the group by passing said request through the remote queue proxy on the first computer and the remote member queue proxy on said second computer.

23. The method of claim 22 wherein providing the application programming interface includes providing software objects implementing said interface that are implemented using object oriented programming principles.

24. The method of claim 22 wherein providing the remote queue proxy includes providing a software object implementing said proxy.

25. A method for passing messages between processes in a distributed system comprising:

providing an application programming interface to processes hosted on computers of the distributed system;

passing a first message from a first process to a second process hosted on one computer of the distributed system, including passing said message through a shared memory accessible to both the first process and the second process; and

passing a second message from the first process to a third process hosted on a second computer of the distributed system, including passing said message over a communication channel coupling the first and the second computers.

26. The method of claim 22 wherein the first process uses the same application programming interface to pass the first message and the second message.

27. The method of claim 22 wherein the first process is unaware of whether the first message and the second message are passing to a process hosted on the first computer or the second computer.

28. The method of claim 22 wherein providing the application programming interface includes providing a queuing interface for passing messages between computers.

29. The method of claim 22 further comprising:

providing a macro queue associated with the plurality of queues; and

wherein passing the first message from the first process to the second process includes calling the application programming interface to initiate a request to the macro queue to obtain a message stored in one of the plurality of queues without identifying a particular queue and selecting a queue from among the plurality of queues and selecting a message from the selected queue.

30. The method of claim 22 further comprising:

providing a remote queue proxy for establishing the communication channel between the first and the second computers.

31. A method for message passing in a distributed system comprising:

providing a queue manager on each of a group of computers in the distributed system;

providing an application programming interface to processes on each of the computers of the group, including providing an interface to accept and to provide messages for passing between processes hosted on the computers;

collecting operational statistics at each of the queue managers related to passing of messages between processes using the application programming interface; and

optimizing passing of the messages according to the collected statistics.

32. A method for fault-tolerant operation of a system comprising:

providing redundant processes for processing messages;

providing a separate replicated message queue for each of the redundant processes;

accepting a message for processing by each of the redundant processes;

enqueuing the message into each of the replicated message queues such that the order of message dequeuing form said queues by the redundant processes is synchronized.

33. The method of claim 22 wherein enqueuing the message into each of the message queues includes performing a logically atomic enqueuing operation on all the queues.

34. The method of claim 22 wherein providing each of said replicated queues includes providing a replicated macro queue associated with a plurality of replicated member queues of said macro queue.

35. A method of managing messages, comprising:

providing an application programming interface (API) to allow a producer module to send a message to a macro queue that manages a plurality of member queues, the API sending the message to the macro queue without identifying one of the plurality of member queues; and

using the same API to allow the producer module to send a message to an individual queue.

36. The method of claim 35 wherein the macro queue selects one or more of the member queues according to a predefined criteria.

37. The method of claim 36 wherein the macro queue, the member queues, and the individual queue are implemented as software objects according to object oriented programming principles.

38. A method of managing messages, comprising:

providing an application programming interface (API) to allow a consumer module to retrieve a message from a macro queue that manages a plurality of member queues, the API retrieving the message from the macro queue without identifying one of the plurality of member queues; and

using the same API to allow the consumer module to retrieve a message from an individual queue.

39. The method of claim 38 wherein the macro queue selects one of the member queues according to a predefined criteria and selects a message from the selected member queue.

40. The method of claim 39 wherein the macro queue, the member queues, and the individual queue are implemented as software objects according to object oriented programming principles.